SSD-TR-91-28

AEROSPACE REPORT NO.
TR-0091(6925-05)-6

AD-A240 410

A Fast Algorithm for Plotting Antenna and Scattering Patterns in Three Dimensions

Prepared by

T. J. PETERS
Communications Systems Subdivision

31 August 1991

Prepared for

SPACE SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
Los Angeles Air Force Base
P. O. Box 92960
Los Angeles, CA 90009-2960

Engineering and Technology Group

91-10511

THE AEROSPACE CORPORATION
El Segundo, California

91 9 12 123

This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-88-C-0089 with the Space Systems Division, P. O. Box 92960, Los Angeles, CA 90009-2960. It was reviewed and approved for The Aerospace Corporation by J. M. Straus, Principal Director, Communications Systems Subdivision.

Seth Parkoff, Capt, USAF, was the project officer for the Mission-Oriented Investigation and Experimentation (MOIE) program.This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

Seth Parkoff

SETH PARKOFF, Capt, USAF
MOIE Project Officer
SSD/MHE

Jonathan M. Emmes

JONATHAN M. EMMES, Maj, USAF
MOIE Program Manager
PL/WCO OL-AH

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) TR-0091(6925-05)-6	5. MONITORING ORGANIZATION REPORT NUMBER(S) SSD-TR-91-28
6a. NAME OF PERFORMING ORGANIZATION The Aerospace Corporation Communications Systems Subdivision 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Space Systems Division
6c. ADDRESS (City, State, and ZIP Code) P. O. Box 92957 Los Angeles, CA 90009-2957	7b. ADDRESS (City, State, and ZIP Code) Los Angeles Air Force Base P. O. Box 92960 Los Angeles, CA 90009-2960
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F04701-88-C-0089
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

A Fast Algorithm for Plotting Antenna and Scattering Patterns in Three Dimensions

12. PERSONAL AUTHOR(S)

Peters, T. J.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM ______ TO ______	31 August 1991	59

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Three-dimensional antenna patterns Three-dimensional graphing methods

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

An algorithm is presented for plotting antenna and scattering patterns in three dimensions on video displays or laser printers. The algorithm exploits the property of single-valued surfaces to allow the implicit removal of hidden lines with virtually no extra computation. This reduces the computation time significantly over that required by more general surface-representation methods. The algorithm is flexible enough to implement on most graphic systems. Simple language-independent pseudo-code is presented and tested for functions in rectangular, cylindrical, and spherical coordinates.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Contents

Figures

Tables

I. Introduction

The representation of antenna and scattering patterns in three dimensions provides a useful tool for analyzing power flow or field strength, especially when used in conjunction with field line contour plots (Ref. 1). Geometrically, these patterns represent surfaces. A comprehensive bibliography of surface-representation algorithms is given by Griffiths (Ref. 2). Most of these algorithms are geared towards representing the complex shape of a physical object. Specific algorithms for plotting mathematical functions of two variables in rectangular coordinates generally follow the method proposed by Wright (Ref. 3). Advanced line drawing algorithms suitable for cylindrical and spherical coordinates have been developed by Scott (Ref. 4). Unfortunately, all these methods require a significant amount of additional computation to be able to plot the surface with the hidden lines removed. The algorithm developed and presented in this paper avoids this extra computation by exploiting known properties of the surfaces being plotted.

In general, line drawing algorithms do not remove hidden lines, but rather, simply do not draw them. An alternative way to remove the hidden lines of a surface is to paint over the hidden part with the same color as the background. This is exactly the technique an artist would use to paint a landscape. The image is placed on the viewing surface from background to foreground and hidden lines are painted over. Of course, this requires the graphics system to be able to fill or erase a polygonal region. Therefore, the algorithm proposed in this paper is not suitable for representing surfaces by means of mechanical pen plotters. However, it is ideally suited for video displays and laser printers.

The algorithm is based on the following postulate. If a function $f(u, v)$, where u and v are two coordinates of an orthogonal system, generates a single-valued surface in the variables u and v, then, there exists a systematic, although not unique, ordered sequence in which to draw the surface from

back to front. This sequence is known, a priori, once the observation angles are specified. Therefore, no hidden line removal is necessary. Thus, plotting a function with the hidden lines removed takes the same amount of time as plotting without removing the hidden lines. A single-valued surface is defined as a surface in which there is a one to one correspondence between each pair of coordinates (u, v) and a point on the surface. Aperture distributions, antenna patterns, and scattering patterns are known to generate surfaces that are single-valued with respect to a particular coordinate system.

II. Coordinate Systems

The functions of interest in this study are plotted in rectangular, cylindrical, and spherical coordinates. The standard variables that describe these coordinate systems are defined by

$$r = \sqrt{x^2 + y^2} \tag{1}$$

$$R = \sqrt{r^2 + z^2} \tag{2}$$

$$\phi = \tan^{-1}(y/x) \tag{3}$$

$$\theta = \tan^{-1}(r/z)\,. \tag{4}$$

In order to organize the algorithm inputs consistently, let $f(u, v)$ describe a function of two variables from one of these coordinate systems. Table 1 shows the convention adopted for the relationship between u and v and the standard variables. Since the final plot is placed on a two-dimensional surface, it is convenient to perform this step in rectangular coordinates. Therefore, all plotting points will be converted to rectangular coordinates prior to any drawing. The conversion conventions are shown in Table 2. Once the points on the surface are converted to rectangular coordinates, the graphical operations of scale, rotation, and projection may be applied.

Table 1. Variable Definitions

	rectangular	cylindrical		spherical
u	x	ϕ	ϕ	ϕ
v	y	r	z	θ

Table 2. Conversions to Rectangular Coordinates

	rectangular	cylindrical		spherical
x	u	$v\cos u$	$f(u,v)\cos u$	$f(u,v)\sin v\cos u$
y	v	$v\sin u$	$f(u,v)\sin u$	$f(u,v)\sin v\sin u$
z	$f(u,v)$	$f(u,v)$	v	$f(u,v)\cos v$

III. Scale, Rotation, and Projection

The surface defined by $f(u,v)$ is represented by the graphical transformations of scale, rotation and projection. Each graphical operation must be applied to individual coordinates of the surface. Let each point of the surface be defined by the vector $\mathbf{w}$ such that

$$\mathbf{w} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{5}$$

The graphical operations may now be defined as matrices that operate on this vector.

In order to enhance some visual attributes, it may be desirable to scale each point before plotting. A scaling matrix $\mathbf{S}$ is defined as

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \tag{6}$$

where each component is some specified constant. Note that to preserve the linearity of the scale, the condition $S_x = S_y$ must be satisfied in cylindrical coordinates and $S_x = S_y = S_z$ must be satisfied in spherical coordinates. The coordinate points are then scaled by forming the matrix vector product $\mathbf{Sw}$. Another type of scaling, which is quite commonly used in plotting antenna and scattering patterns, involves converting the function to decibels. This allows the viewer to observe more detail of the sidelobe behavior. Let f be normalized to the range $0 \le f \le 1$. Then define the zero reference in dB as ν. Next, define a plot floor level in dB as η, such that $\eta \ge \nu$. This plot floor is the level to which the function is set to for any value below the floor. This avoids a cluttered graph that results from too many low-level sidelobes. The function f can then be converted to a dB scale such that $0 \le f_{\mathrm{dB}} \le 1$ by the nonlinear transformation

$$f_{\mathrm{dB}} = \begin{cases} \frac{1}{|\nu|}(\eta + |\nu|) & f \le 10^{(\eta/10)} \\ \frac{1}{|\nu|}(10 \log(f) + |\nu|) & f > 10^{(\eta/10)}. \end{cases} \tag{7}$$

The observer is assumed to be stationary, so the surface of the function must be rotated to the correct view. Viewing any finite three-dimensional object requires a minimum of one rotation axis. However, it is usually necessary to have rotation around two axes. These axes should be perpendicular to allow the widest range of viewing angles. The convention adopted in this study is to allow an azimuthal rotation around the z axis and an elevation rotation in the yz plane. This is conveninent for plotting antenna patterns, aperture distributions and scattering patterns. The visual result is a graph that appears to spin in azimuth around the z axis and is tipped in elevation toward or away from the viewer.

It is important to distinguish between the observation angles and the rotation angles. The viewer-supplied elevation observation angle is defined as θ_0, and the azimuth observation angle is defined as ϕ_0. These angles will be restricted to the ranges $0 \leq \theta_0 \leq \pi$ and $0 \leq \phi_0 \leq 2\pi$, respectively. The use of these angles provides viewers a familiar frame of reference. In order to view the surface from these angles, it is necessary to define an azimuth rotation angle α and an elevation rotation angle β. These angles are dependent on the observation angles and the orientation of the viewer to the rectangular coordinate system. It is assumed that the observer will look in from the positive z axis onto the xy plane. The x axis increases from left to right and the y axis increases from bottom to top. Once this convention is established and the observation angles are specified, the rotation angles can be calculated as

$$\alpha = -\frac{\pi}{2} - \phi_0 \tag{8}$$

$$\beta = \theta_0. \tag{9}$$

The order of rotation is not commutative. The plots must first be spun in azimuth by the angle α and then tipped in elevation by the angle β. Reversing the order would allow the plot to tip from side to side, which presents an awkward picture. The azimuth rotation in the xy plane is represented by the

matrix vector product $\mathbf{R}_\alpha \mathbf{w}$, where

$$\mathbf{R}_\alpha = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

and the elevation rotation in the yz plane by the product $\mathbf{R}_\beta \mathbf{w}$, where

$$\mathbf{R}_\beta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix}. \tag{11}$$

The complete transformation can be represented by the matrix $\mathbf{T}$, defined by

$$\mathbf{T} = \mathbf{R}_\beta \mathbf{R}_\alpha \mathbf{S}. \tag{12}$$

If $\mathbf{w}'$ represents the transformed coordinates then

$$\mathbf{w}' = \mathbf{T}\mathbf{w}. \tag{13}$$

Viewing a three-dimensional plot on a two-dimensional surface requires a projection of the three-dimensional coordinate points onto a two-dimensional viewing plane. Perspective projection implies that the further a point is away from the viewer, the smaller it appears. This type of projection is appropriate for images that evoke a strong depth cue such as a building or a landscape. However, mathematical functions do not require a strong depth cue since there is no physical object being represented. Therefore, it is sufficient to project each point along a parallel line until it intercepts the viewing plane. This type of projection is called parallel projection. Using parallel projection implies that only the x' and y' components of the vector $\mathbf{w}$ are needed to represent the surface. The z' coordinate represents the depth of each point and is not used.

These three graphical transformations are independent of the type of function being plotted. They represent the transformation of a point on the function surface to a point on the viewing surface. The order in which points are operated on by the transformation is determined by the sequencing algorithm discussed in the next section.

IV. Algorithm Description

The proposed algorithm is based on the generic "painter's algorithm." This concept means that the surface of the plot is rendered by building up the image from back to front. Since the function has two variables, the surface is most naturally described by a collection of quadrilaterals. Each quadrilateral is placed on the viewing surface and is painted with the background color of the surface. A line around the perimeter is then drawn. The key, of course, is to know the order in which to place the quadrilaterals on the viewing surface. A previous method, developed by the author (Ref. 5), used the transformed z coordinate of the centroid of each quadrilateral and sorted by depth. This method was general enough to plot any surface. However, the time required to compute the depth of each centroid and to sort the values led to a significant time delay. The new method proposed avoids this delay by drawing in a prescribed order based on the observation angles ϕ_0 and θ_0.

The variables u and v are discretized such that $u = u_m$ for $m = 1...N_u$ and $v = v_n$ for $n = 1...N_v$. By convention, the values sequence from the minimum to the maximum values. The function $f(u, v)$ is then sampled at the discrete points $f_{m,n} = f(u_m, v_n)$. Each quadrilateral has a reference corner that has index values (m, n). The other three corners are dependent on these indices and are given by $(m+1, n)$, $(m+1, n+1)$, and $(m, n+1)$. It should be noted that the definition of a quadrilateral is extended to allow any number of corners to have the same location. Therefore, a point, a line, and a triangle, can also be represented by a quadrilateral. The crux of the algorithm is to find a systematic method of sequencing through the indices in order to approximate a back-to-front ordering. The sequencing orders presented for the three coordinate systems analyzed were chosen based on ease of programming.

V. Rectangular Coordinates

Rectangular coordinates are the most common way of representing patterns in three dimensions. The coordinates x and y are replaced by ϕ and θ. Assume for simplicity that the coordinates are translated so that the origin is in the interior of the range. This does not affect the generality of the algorithm, but merely the presentation. Since the function is plotted on a rectangular base, it may be surmised that one corner of the surface will always be nearest to the observer and the opposite corner will be the farthest away. Because of the rotation conventions used, this is dependent only on the azimuth observation angle ϕ_0 and independent of the elevation observation angle θ_0. Therefore, the first part of the algorithm determines which corner is nearest to the observer. Once the orientation is established, the quadrilaterals are simply drawn from the back corner to the front corner by rows that alternate in direction. Alternating the rows helps to move forward in a more uniform manner. Figure 1 shows the drawing directions as the quadrilaterals are placed on the viewing surface from back to front. Note that at the angles 0°, 90°, 180°, 270° and 360° there are two back corners equidistant from the viewer. In this case, it does not matter which corner is chosen as long as it is chosen consistently. Figure 2 shows the far-field power pattern of a uniformly excited square aperture plotted at observation angles $\phi_0 = 30°$ and $\theta_0 = 60°$. The plotting algorithm used is given in pseudo-code as follows:

```
M = minimum of (N_u, N_v)
if 0 ≤ φ0 < π/2 then {find nearest corner}
    i = 1,   j = 1
else if π/2 ≤ φ0 < π then
    i = −1,   j = 1
else if π ≤ φ0 < 3π/2 then
    i = −1,   j = −1
else if 3π/2 ≤ φ0 ≤ 2π then
    i = 1,   j = −1
end if
loop from l = 1 to M − 1
    if 0 ≤ φ0 < π/2 then {find nearest corner}
```

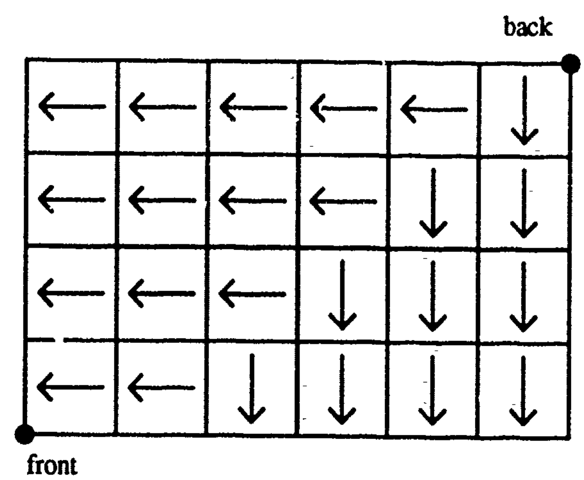

Fig. 1. Drawing flow pattern for rectangular coordinates

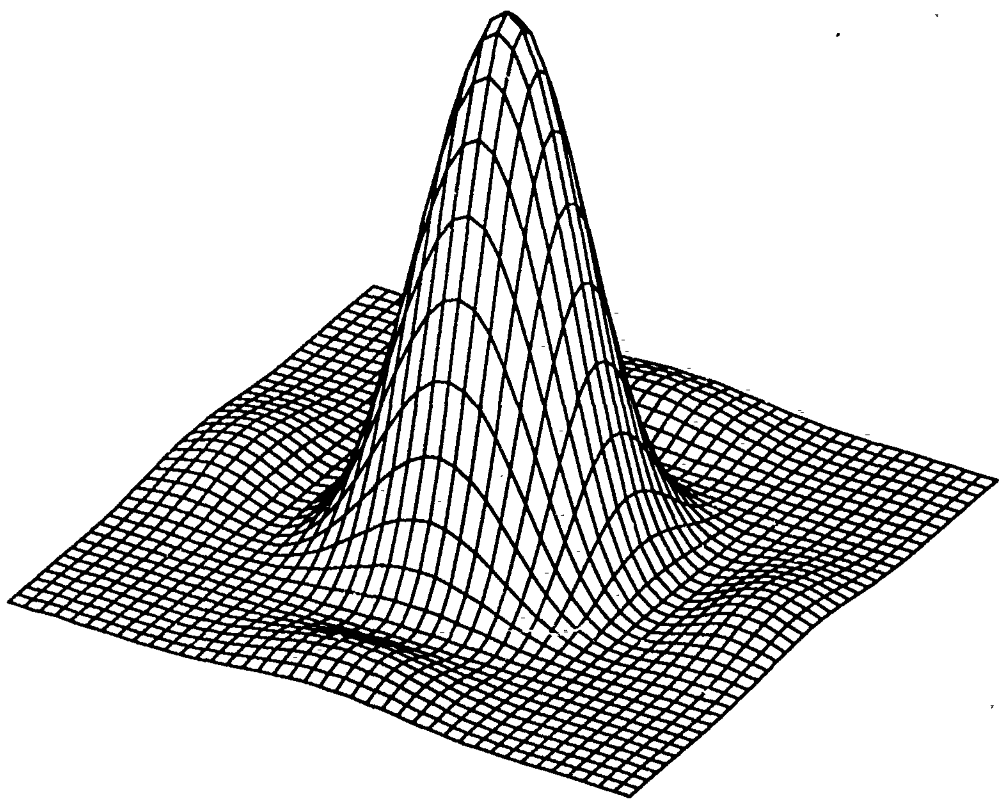

Fig. 2. Power pattern of uniformly excited square aperture viewed from $\theta_0 = 60°$ and $\phi_0 = 30°$

$m_0 = l - 1, \quad n_0 = l$
$m_1 = l, \quad n_1 = l$
else if $\pi/2 \le \phi_0 < \pi$ **then**
$m_0 = N_u - l + 1, \quad n_0 = l$
$m_1 = N_u - l, \quad n_1 = l$
else if $\pi \le \phi_0 < 3\pi/2$ **then**
$m_0 = N_u - l + 1, \quad n_0 = N_v - l$
$m_1 = N_u - l, \quad n_1 = N_v - l$
else if $3\pi/2 \le \phi_0 \le 2\pi$ **then**
$m_0 = l - 1, \quad n_0 = N_v - l$
$m_1 = l, \quad n_1 = N_v - l$
end if
loop from $k = 1$ **to** $N_u - l - 1$
$m = m_0 + ik, \quad n = n_0$
{ get (u_m, v_n) and other 3 corners }
{ convert to rectangular coordinates - Table 2 }
{ scale, rotate, and project using Eqn. (13)}
{ fill quadrilateral, then draw perimeter }
continue k **loop**
loop from $k = 1$ **to** $N_v - l$
$m = m_1, \quad n = n_1 + jk$
{ get (u_m, v_n) and other 3 corners }
{ convert to rectangular coordinates - Table 2 }
{ scale, rotate, and project using Eqn. (13)}
{ fill quadrilateral, then draw perimeter }
continue k **loop**
continue l **loop**

VI. Cylindrical Coordinates

Cylindrical coordinate functions of the form $z = f(\phi, r)$ and $r = f(\phi, z)$ are commonly encountered in antenna and scattering analysis. Far-field patterns can be plotted in the coordinates $f(\phi, r)$ by letting $r = \theta$. This type of plot is good for observing the finer details of the sidelobe structure. It is also a natual way to plot circular aperture distributions. Alternatively, functions of the form $r = f(\phi, z)$ are useful for observing near-field patterns or surface currents on structures such as a cylinder or a body of revolution.

Cylindrical functions of the form $z = f(\phi, r)$ are plotted on a circular base with a specified foreground angle ϕ_0 and a background angle $\phi_a = \phi_0 \pm \pi$, chosen such that $0 \leq \phi_a \leq 2\pi$. Therefore, the ϕ dependency of the quadrilaterals should be drawn starting with the background angle ϕ_a and procceeding toward the foreground observation angle ϕ_0. The method employed is to find the index m of the u_m nearest to ϕ_a and then alternately increase and decrease the value to draw from back to front. For simplicity, assume that $u_{min} = 0$ and $u_{max} = 2\pi$. Since ϕ is periodic, the index m must be periodic with period $N_u - 1$. Drawing the radial dependence from back to front is dependent on the value of ϕ. Observation of the drawing flow pattern in Fig. 3 shows that the radial dependence should be drawn from r_{max} to r_{min} when the angle u_m is greater than 90° from ϕ_0, and from r_{min} to r_{max} when the angle is less than 90° from ϕ_0. Figure 4 shows the power pattern of a uniformly excited square aperture plotted in dB, with $\nu = -40$ dB and $\eta = -25$ dB. In some plotting situations it is desirable to remove an angular sector, to allow better viewing of the surface. This can be easily accomplished because of the way the ϕ values are alternated when plotting. Defining N_s as the number of segments to remove, the total number of ϕ values is just reduced by that amount. Figure 5 illustrates the use of angular cuts in a plot of the magnitude of the far-field electric field of a uniformly excited circular aperture. The following pseudo-code implements the algorithm:

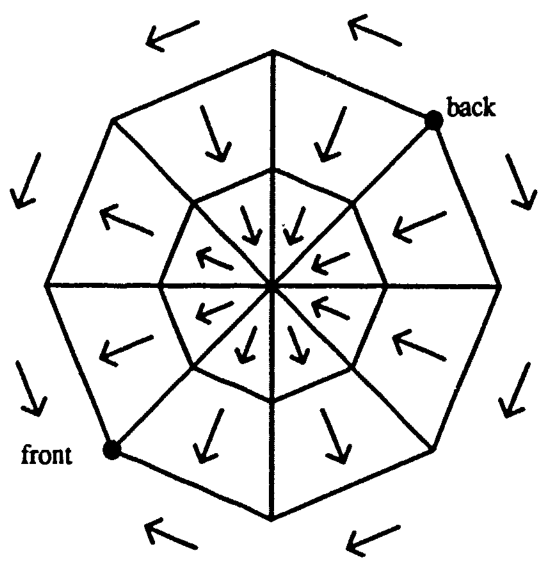

Fig. 3. Drawing flow pattern for cylindrical coordinates

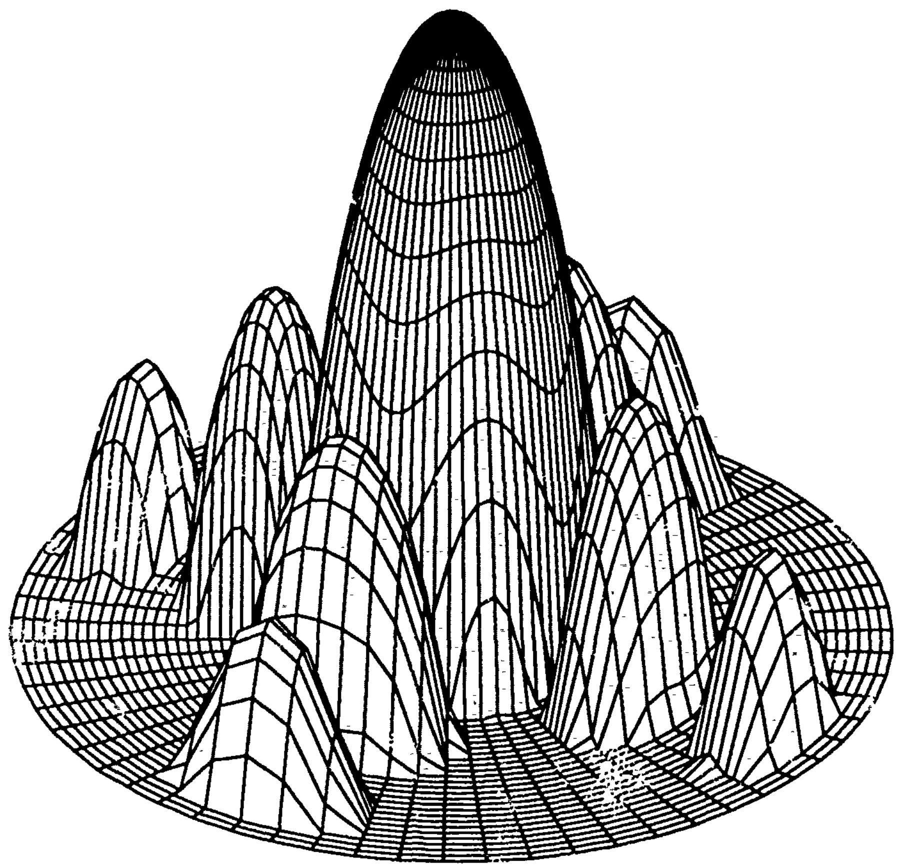

Fig. 4. Power pattern of uniformly excited square aperture viewed from $\theta_0 = 60°$ and $\phi_0 = 30°$ and plotted in dB, with $\nu = -40$ dB and $\eta = -25$ dB

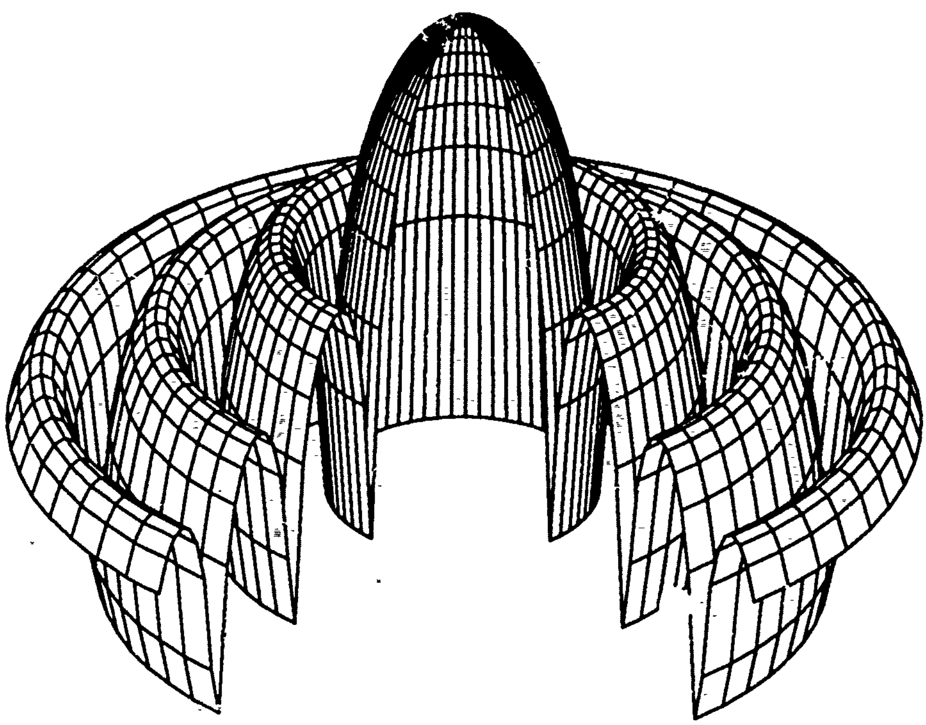

Fig. 5. Magnitude of far-field electric field of a uniformly excited circular aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$ and plotted in dB, with $N_s = 14$, $\nu = -40$ dB and $\eta = -25$ dB

loop from $k = 1$ **to** $N_u - 1$ {find index of angle ϕ_a}
  if $\phi_a \geq u_k$ **and** $\phi_a \leq u_{k+1}$ **then**
    $m = k$
  end if
continue k **loop**
loop from $l = 1$ **to** $N_u - N_s - 1$
  $m = m - (-1)^l(l-1)$
  if $(m < 1)$ **then** {force index to be periodic}
    $m = m + N_u - 1$
  else if $(m > N_u - 1)$ **then**
    $m = m - (N_u - 1)$
  end if
  if $|u_m - \phi_0| < \pi/2$ **or** $|u_m - \phi_0| > 3\pi/2$ **then**
    $n_0 = 0, \quad i = 1$ {draw from r_{min} to r_{max}}
  else
    $n_0 = N_v, \quad i = -1$ {draw from r_{max} to r_{min}}
  end if
  loop from $k = 1$ **to** $N_v - 1$
    $n = n_0 + ik$
    {get (u_m, v_n) and other 3 corners }
    {convert to rectangular coordinates - Table 2 }
    {scale, rotate, and project using Eqn. (13)}
    {fill quadrilateral, then draw perimeter }
  continue k **loop**
continue l **loop**

The algorithm for plotting functions of the form $r = f(\phi, z)$ uses the same procedure for handling the ϕ variation. The z variation is handled very easily by simply noting that if the observation angle θ_0 is less than or equal to 90°, then draw from z_{min} to z_{max}. Otherwise, if θ_0 is greater than 90°, then draw from z_{max} to z_{min}. Figure 6 shows a section of circular waveguide and the magnitude of the longitudinal surface current induced on the inner walls for the TM_{21} mode. The pseudo-code for the algorithm is given as follows:

loop from $k = 1$ **to** $N_u - 1$ {find index of angle ϕ_a}
  if $\phi_a \geq u_k$ **and** $\phi_a \leq u_{k+1}$ **then**
    $m_a = k$
  end if
continue k **loop**
loop from $k = 1$ **to** $N_v - 1$
  if $\theta_0 \leq \pi/2$ **then**
    $n = k$ {draw from z_{min} to z_{max}}
  else
    $n = N_v - k$ {draw from z_{max} to z_{min}}
  end if
  $m = m_a$

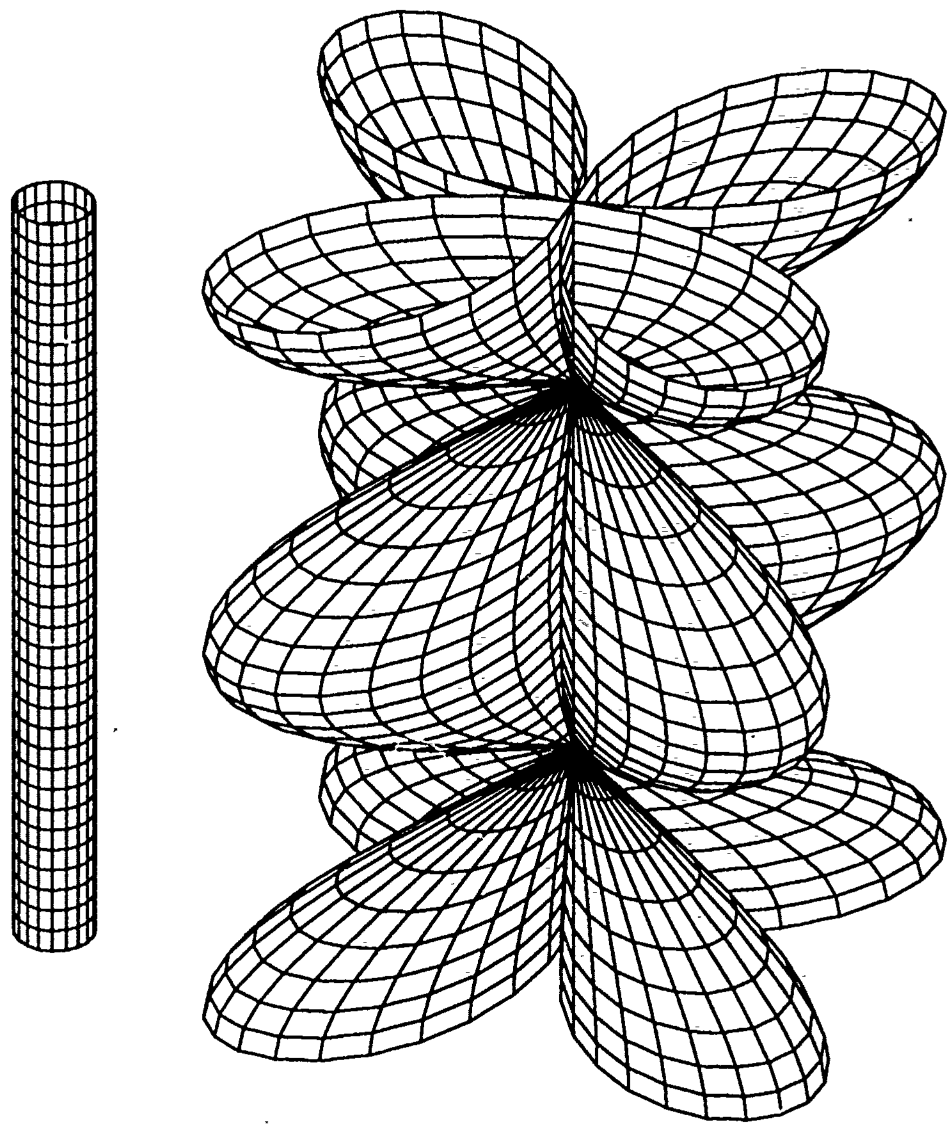

Fig. 6. Circular waveguide and magnitude of surface current induced on inside wall

```
    loop from l = 1 to N_u - N_s - 1
        m = m - (-1)^l (l - 1)
        if (m < 1) then    {force index to be periodic}
            m = m + N_u - 1
        else if (m > N_u - 1) then
            m = m - (N_u - 1)
        end if
        {get (u_m, v_n) and other 3 corners }
        {convert to rectangular coordinates - Table 2 }
        {scale, rotate, and project using Eqn. (13)}
        {fill quadrilateral, then draw perimeter }
    continue l loop
continue k loop
```

VII. Spherical Coordinates

The flow of power in far-field antenna and scattering patterns is best described by using plots in spherical coordinates. Establishing the front and back orientation requires knowledge of both observation angles. The azimuth background angle ϕ_a is defined the same as in cylindrical coordinates. Therefore, the ϕ variation is drawn in exactly the same way as for cylindrical plots. The elevation background angle is defined as $\theta_a = \pi - \theta_0$. As shown in Fig. 7, the elevation dependence is drawn in opposite directions away from θ_a when u_m is greater than 90° from ϕ_0. When u_m is less than 90° from ϕ_0, the elevation dependence is drawn from the minimum and maximum limits to θ_0. Figure 8 shows a power pattern of a circular aperture plotted in dB with $\nu = \eta = -60$ dB. As with cylindrical coordinates, it is sometimes convenient to cut away an angular piece of the plot so that the detail of the sidelobe pattern may be observed. However, since the lobes of spherical plots are generally closed surfaces, it looks better if the angular cut is filled with the background color. This is illustrated in Fig. 9. Filling this cut requires making a polygon using all the θ values at a constant ϕ. The following pseudo-code implements the algorithm:

loop from $k = 1$ **to** $N_u - 1$ {find index of angles ϕ_0,ϕ_a}
    if $\phi_0 \geq u_k$ **and** $\phi_0 \leq u_{k+1}$ **then**
        $m_0 = k$
    end if
    if $\phi_a \geq u_k$ **and** $\phi_a \leq u_{k+1}$ **then**
        $m_a = k$
    end if
continue k **loop**
$n_0 = N_v \quad n_a = N_v$
loop from $k = 1$ **to** $N_v - 1$ {find index of angles θ_0, θ_a}
    if $\theta_0 \geq v_k$ **and** $\theta_0 \leq v_{k+1}$ **then**
        $n_0 = k$
    end if
    if $\theta_a \geq v_k$ **and** $\theta_a \leq v_{k+1}$ **then**
        $n_a = k$
    end if
continue k **loop**
$m = m_a$

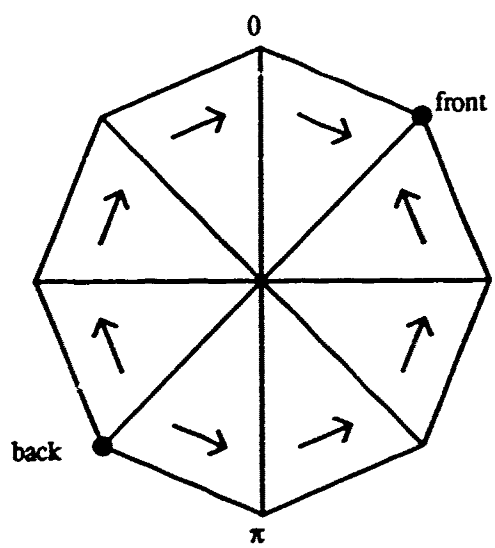

Fig. 7. Elevation dependence drawing pattern for spherical coordinates

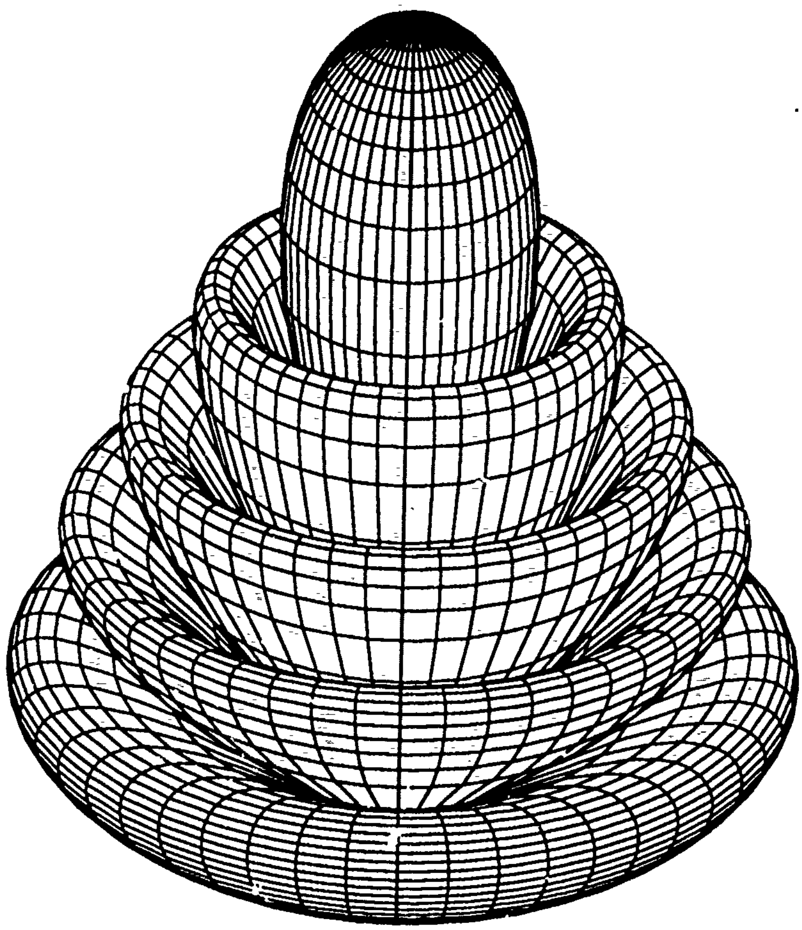

Fig. 8. Power pattern of uniformly excited circular aperture viewed from $\theta_0 = 60°$ and $\phi_0 = 30°$

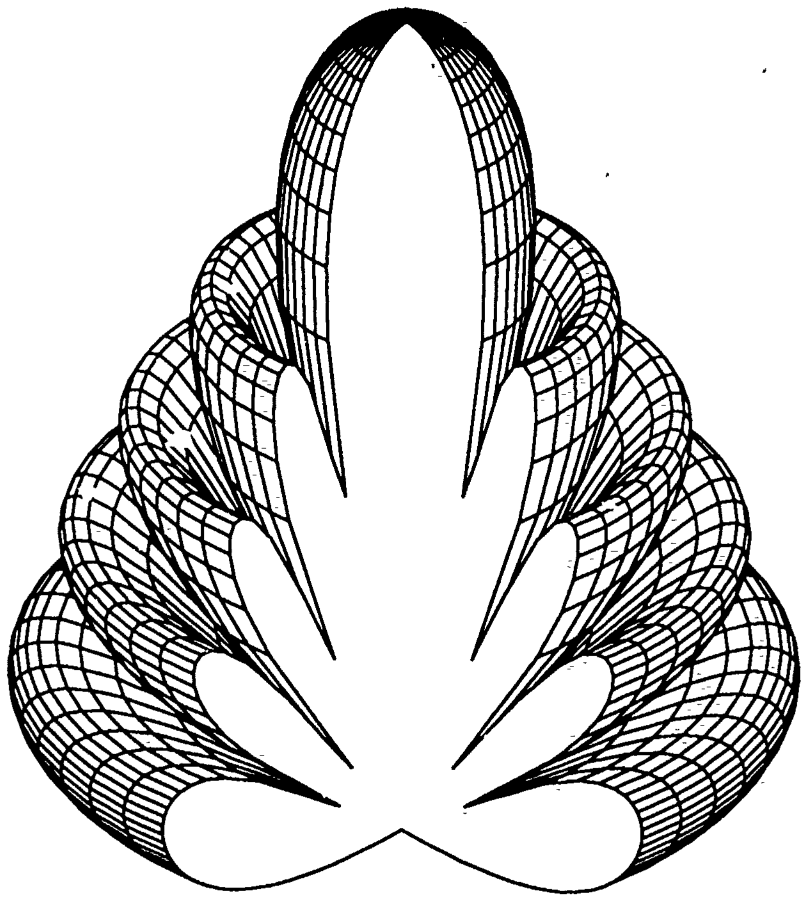

Fig. 9. Power pattern of uniformly excited circular aperture viewed from $\theta_0 = 60°$ and $\phi_0 = 30°$

```
loop from $l = 1$ to $N_u - N_s - 1$
    $m = m - (-1)^l (l - 1)$
    if ($m < 1$) then    {force index to be periodic}
        $m = m + N_u - 1$
    else if ($m > N_u - 1$) then
        $m = m - (N_u - 1)$
    end if
    if $|u_m - \phi_0| < \pi/2$ or $|u_m - \phi_0| > 3\pi/2$ then
        $i = N_v, j = 0, L_1 = N_v - n_0, L_2 = n_0 - 1$
    else
        $i = M_a, j = n_a - 1, L_1 = n_a - 1, L_2 = N_v - n_a$
    end if
    loop from $k = 1$ to $L_1$
        $n = i - k$
        { get $(u_m, v_n)$ and other 3 corners }
        { convert to rectangular coordinates - Table 2 }
        { scale, rotate, and project using Eqn. (13)}
        { fill quadrilateral, then draw perimeter }
    continue $k$ loop
    loop from $k = 1$ to $L_2$
        $n = j + k$
        { get $(u_m, v_n)$ and other 3 corners }
        { convert to rectangular coordinates - Table 2 }
        { scale, rotate, and project using Eqn. (13)}
        { fill quadrilateral, then draw perimeter }
    continue $k$ loop
    if $l > N_u - N_s$ - 3 and $N_s > 0$ then
        if $u_m \geq \phi_0$ then
            if ($u_m \geq \gamma_0$ and $u_m \leq \phi_a$) then
                $p = m$
            else
                $p = m + 1$
            end if
        else
            if ($u_m \geq \phi_a$ and $u_m \leq \phi_0$) then
                $p = m + 1$
            else
                $p = m$
            end if
        loop from $q = 1$ to $N_v$
        { form polygon with vertices $(u_p, v_q)$ }
        continue $q$ loop
        { fill polygon, then draw perimeter }
    end if
continue $l$ loop
```

VIII. Conclusion

An algorithm has been presented that allows the rapid plotting of antenna and scattering patterns in three dimensions. Because of the special properties of the types of functions considered, the plotting speed is essentially the same as if no hidden lines were removed Any graphics system that allows a polygon fill operation may implement the algorithm. Although not suitable for mechanical pen plotters, the algorithm is ideal for video displays and laser printers. The specific variations in rectangular, cylindrical, and spherical coordinates have been developed and tested yielding excellent results.

References

1. K. W. Kark and R. Dill, "A General Theory on the Graphical Representation of Antenna-Radiation Fields," IEEE Trans. Antennas Propagat., vol. AP-38, no. 2, pp.160-165, Feb. 1990.

2. J. G. Griffiths, "A Bibliography of Hidden-Line and Hidden Surface Algoritlms," Computer Aided Design, 10(3), May 1978, pp. 203-206.

3. T. J. Wright, "A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables," IEEE Trans. Comput., vol. C-19, pp.28-33., January 1973.

4. W. R. Scott, Jr., "A General Program for Plotting Three-Dimensional Antenna Patterns," in 1987 IEEE Antennas Propagat. Soc. Symp. Dig., vol. 1, June 1988, pp. 330-333.

5. T. J. Peters, "Computation of the Scattering by Planar and Non-Planar Plates Using a Conjugate Gradient FFT Method," Ph.D. dissertation, Radi. ion Laboratory, The University of Michigan, 1988, pp. 206-221.

Appendix: Fortran 77 Programs

This appendix contains the Fortran 77 programs RECT3D, CYLA3D, CYLR3D, and SPHR3D as well as all associated subroutines. The required inputs to each program are listed in the comments found in the source code. These programs were used to generate all the results presented in this report. The output of each program is a PostScript file. Information on programming in the PostScript language is available from most bookstores.

```
      PROGRAM RECT3D
C    *******************************************************************
C    * THIS PROGRAM PRODUCES A POSTSCRIPT FILE REPRESENTING A THREE    *
C    * DIMENSIONAL PLOT OF A FUNCTION IN RECTANGULAR COORDINATES.      *
C    *******************************************************************
C    * TIMOTHY J. PETERS                             LAST UPDATED      *
C    * THE AEROSPACE CORPORATION                     3/1/91            *
C    * 2350 EAST EL SEGUNDO BOULEVARD.                                 *
C    * EL SEGUNDO, CA 90245                                            *
C    *******************************************************************
C    * INPUTS:                                                         *
C    *                                                                 *
C    * NU   - NUMBER OF U POINTS.                                      *
C    * NV   - NUMBER OF V POINTS.                                      *
C    * AX  - X DIRECTION SCALE FACTOR.                                 *
C    * AY  - Y DIRECTION SCALE FACTOR.                                 *
C    * AZ  - Z DIRECTION SCALE FACTOR.                                 *
C    * PO  - PHI OBSERVATION ANGLE IN RANGE 0<=PO<=2PI.                *
C    * TO  - THETA OBSERVATION ANGLE IN RANGE 0<=PO<=PI.               *
C    * IC  - IF IC=1 THEN CONVERT THE FUNCTION VALUES TO DB.           *
C    *       NOTE THAT IF IC=1 THEN F MUST BE IN THE RANGE 0<=F<=1.    *
C    * ETA - PLOT FLOOR IN DB.                                         *
C    * NUU - EFFECTIVE ZERO IN DB.                                     *
C    * U(MAX) - U COORDINATE ARRAY.                                    *
C    * V(MAX) - V COORDINATE ARRAY.                                    *
C    * F(MAX,MAX) - FUNCTION VALUE MATRIX.                             *
C    *                                                                 *
C    * OUTPUT:                                                         *
C    *                                                                 *
C    * RECT.PS  - POSTSCRIPT FILE REPRESENTING THE PLOT.               *
C    *                                                                 *
C    *******************************************************************
      PARAMETER (MAX=100)
      REAL*4 U(MAX),V(MAX),F(MAX,MAX),NUU
      OPEN(UNIT=2,FILE='RECT.PS')
      REWIND(2)
      RAD=.17453293E-01
      PI=.3141593E+01
C    *******************************************************************
C    * READ THE INPUTS.                                                *
C    *******************************************************************
      OPEN(UNIT=1,FILE='DATA')
      READ(1,*) NU,NV,AX,AY,AZ,PO,TO,IC,ETA,NUU
      READ(1,*) (U(M),M=1,NU)
      READ(1,*) (V(N),N=1,NV)
      READ(1,*) ((F(M,N),M=1,NU),N=1,NV)
C    *******************************************************************
C    * DEFINE SOME MACROS IN POSTSCRIPT.                               *
C    *******************************************************************
```

```
      WRITE(2,100) 'initgraphics erasepage letter'
      WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
      WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
      WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
C     *******************************************************************
C     * SET THE LINE CHARACTERISTICS.                                   *
C     *******************************************************************
      WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
C     *******************************************************************
C     * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAPER.      *
C     *******************************************************************
      XOFFSET=306.0
      YOFFSET=396.0
      WRITE(2,101) XOFFSET,YOFFSET,' t'
C     *******************************************************************
C     * CONVERT THE OBSERVATION ANGLES TO RADIANS.                      *
C     *******************************************************************
      POBS=RAD*PO
      TOBS=RAD*TO
C     *******************************************************************
C     * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION *
C     * ANGLES.                                                         *
C     *******************************************************************
      RP=-POBS-PI/2.0
      RT=TOBS
      CP=COS(RP)
      SP=SIN(RP)
      CT=COS(RT)
      ST=SIN(RT)
C     *******************************************************************
C     * IF REQUESTED CONVERT THE DATA TO DB SCALE.                      *
C     *******************************************************************
      IF (IC .EQ. 1) THEN
        TR=10.0**(0.1*ETA)
        DO 1 M=1,NU
         DO 2 N=1,NV
          IF(F(M,N) .LE. TR) THEN
            F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
          ELSE
            F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
          END IF
  2      CONTINUE
  1     CONTINUE
      ELSE
      END IF
C     *******************************************************************
C     * SET SOME CONSTANTS.                                             *
C     *******************************************************************
      IF ((POBS .GE. 0.0) .AND. (POBS .LT. PI/2.0)) THEN
```

```
          IS1=1
          IS2=1
        ELSE IF ((POBS .GE. PI/2.0) .AND. (POBS .LT. PI)) THEN
          IS1=-1
          IS2=1
        ELSE IF ((POBS .GE. PI) .AND. (POBS .LT. 3.0*PI/2.0)) THEN
          IS1=-1
          IS2=-1
        ELSE IF ((POBS .GE. 3.0*PI/2.0) .AND. (POBS .LE. 2.0*PI)) THEN
          IS1=1
          IS2=-1
        ELSE
        END IF
C      ******************************************************************
C      * BEGIN SEQUENCE.                                                *
C      ******************************************************************
        DO 3 L=1,NV-1
          IF ((POBS .GE. 0.0) .AND. (POBS .LT. PI/2.0)) THEN
            M0=L-1
            N0=L
            M1=L
            N1=L
          ELSE IF ((POBS .GE. PI/2.0) .AND. (POBS .LT. PI)) THEN
            M0=NU-L+1
            N0=L
            M1=NU-L
            N1=L
           ELSE IF ((POBS .GE. PI) .AND. (POBS .LT. 3.0*PI/2.0)) THEN
            M0=NU-L+1
            N0=NV-L
            M1=NU-L
            N1=NV-L
          ELSE IF ((POBS .GE. 3.0*PI/2.0) .AND. (POBS .LE. 2.0*PI)) THEN
            M0=L-1
            N0=NV-L
            M1=L
            N1=NV-L
          ELSE
          END IF
C        ****************************************************************
C        * LOOP THROUGH THE U VALUES WITH V CONSTANT.                   *
C        ****************************************************************
          DO 4 K=1,NU-L
            M=M0+IS1*K
            N=N0
C          **************************************************************
C          * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL.               *
C          **************************************************************
            XS1=U(M)
```

```
         YS1=V(N)
         ZS1=F(M,N)
         XS2=U(M+1)
         YS2=V(N)
         ZS2=F(M+1,N)
         XS3=U(M+1)
         YS3=V(N+1)
         ZS3=F(M+1,N+1)
         XS4=U(M)
         YS4=V(N+1)
         ZS4=F(M,N+1)
C       *************************************************************
C       * ROTATE THE 4 VERTICES OF THE QUADRILATERAL.               *
C       *************************************************************
         CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)
         CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
         CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
         CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
C       *************************************************************
C       * FILL THE QUADRILATERAL.                                   *
C       *************************************************************
         WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
C       *************************************************************
C       * DRAW PERIMETER OF THE QUADRILATERAL.                      *
C       *************************************************************
         WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
    4   CONTINUE
C     ***************************************************************
C     * LOOP THROUGH THE V VALUES WITH U CONSTANT.                  *
C     ***************************************************************
      DO 5 K=1,NV-L-1
         M=M1
         N=N1+IS2*K
C       *************************************************************
C       * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL.              *
C       *************************************************************
         XS1=U(M)
         YS1=V(N)
         ZS1=F(M,N)
         XS2=U(M+1)
         YS2=V(N)
         ZS2=F(M+1,N)
         XS3=U(M+1)
         YS3=V(N+1)
         ZS3=F(M+1,N+1)
         XS4=U(M)
         YS4=V(N+1)
         ZS4=F(M,N+1)
C       *************************************************************
```

```
C         * ROTATE THE 4 VERTICES OF THE QUADRILATERAL.                    *
C         ******************************************************************
             CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)
             CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
             CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
             CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
C         ******************************************************************
C         * FILL THE QUADRILATERAL.                                        *
C         ******************************************************************
             WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
C         ******************************************************************
C         * DRAW PERIMETER OF THE QUADRILATERAL.                           *
C         ******************************************************************
             WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
    5      CONTINUE
    3    CONTINUE
C     **********************************************************************
C     * SHOW THE PAGE.                                                     *
C     **********************************************************************
         WRITE(2,100) 'showpage'
C     **********************************************************************
C     * FORMATS.                                                           *
C     **********************************************************************
  100    FORMAT(A72)
  101    FORMAT(F7.2,1X,F7.2,A58)
  200    FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l c')
  201    FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l w')
         END
C
         SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
         X=AX*COS(RP)*XA-AY*SIN(RP)*YA
         Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
         RETURN
         END
```

```
      PROGRAM CYLA3D
C    ******************************************************************
C    * THIS PROGRAM PRODUCES A POSTSCRIPT FILE REPRESENTING A THREE   *
C    * DIMENSIONAL PLOT OF A FUNCTION IN CYLINDRICAL COORDINATES.     *
C    ******************************************************************
C    * TIMOTHY J. PETERS                               LAST UPDATED   *
C    * THE AEROSPACE CORPORATION                       3/1/91         *
C    * 2350 EAST EL SEGUNDO BOULEVARD.                                *
C    * EL SEGUNDO, CA 90245                                           *
C    ******************************************************************
C    * INPUTS:                                                        *
C    *                                                                *
C    * NU  - NUMBER OF U POINTS.                                      *
C    * NV  - NUMBER OF V POINTS.                                      *
C    * AX  - X DIRECTION SCALE FACTOR.                                *
C    * AY  - Y DIRECTION SCALE FACTOR.                                *
C    * AZ  - Z DIRECTION SCALE FACTOR.                                *
C    * PO  - PHI OBSERVATION ANGLE IN RANGE 0<=PO<=2PI.               *
C    * TO  - THETA OBSERVATION ANGLE IN RANGE 0<=PO<=PI.              *
C    * IC  - IF IC=1 THEN CONVERT THE FUNCTION VALUES TO DB.          *
C    *       NOTE THAT IF IC=1 THEN F MUST BE IN THE RANGE 0<=F<=1.   *
C    * ETA - PLOT FLOOR IN DB.                                        *
C    * NUU - EFFECTIVE ZERO IN DB.                                    *
C    * NS  - NUMBER OF SEGMENTS TO REMOVE.                            *
C    * U(MAX) - U COORDINATE ARRAY.                                   *
C    * V(MAX) - V COORDINATE ARRAY.                                   *
C    * F(MAX,MAX) - FUNCTION VALUE MATRIX.                            *
C    *                                                                *
C    * OUTPUT:                                                        *
C    *                                                                *
C    * CYLA.PS  - POSTSCRIPT FILE REPRESENTING THE PLOT.              *
C    *                                                                *
C    ******************************************************************
      PARAMETER (MAX=100)
      REAL*4 U(MAX),V(MAX),F(MAX,MAX),NUU
      OPEN(UNIT=2,FILE='CYLA.PS')
      REWIND(2)
      RAD=.17453293E-01
      PI=.3141593E+01
C    ******************************************************************
C    * READ THE INPUTS.                                               *
C    ******************************************************************
      OPEN(UNIT=1,FILE='DATA')
      READ(1,*) NU,NV,AX,AY,AZ,PO,TO,IC,ETA,NUU,NS
      READ(1,*) (U(M),M=1,NU)
      READ(1,*) (V(N),N=1,NV)
      READ(1,*) ((F(M,N),M=1,NU),N=1,NV)
C    ******************************************************************
C    * DEFINE SOME MACROS IN POSTSCRIPT.                              *
```

```
C     ******************************************************************
       WRITE(2,100) 'initgraphics erasepage letter'
       WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
       WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
       WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
C     ******************************************************************
C     *.SET THE LINE CHARACTERISTICS.                                  *
C     ******************************************************************
       WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
C     ******************************************************************
C     * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAGE.      *
C     ******************************************************************
       XOFFSET=306.0
       YOFFSET=396.0
       WRITE(2,101) XOFFSET,YOFFSET,' t'
C     ******************************************************************
C     * CONVERT THE OBSERVATION ANGLES TO RADIANS.                     *
C     ******************************************************************
       POBS=RAD*P0
       TOBS=RAD*T0
C     ******************************************************************
C     * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION *
C     * ANGLES.                                                        *
C     ******************************************************************
       RP=-POBS-PI/2.0
       RT=TOBS
       CP=COS(RP)
       SP=SIN(RP)
       CT=COS(RT)
       ST=SIN(RT)
C     ******************************************************************
C     * IF REQUESTED CONVERT THE DATA TO DB SCALE.                     *
C     ******************************************************************
       IF (IC .EQ. 1) THEN
         TR=10.0**(0.1*ETA)
         DO 1 M=1,NU
          DO 2 N=1,NV
           IF(F(M,N) .LE. TR) THEN
             F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
           ELSE
             F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
           END IF
 2        CONTINUE
 1       CONTINUE
       ELSE
       END IF
C     ******************************************************************
C     * DETERMINE THE VALUE WHICH IS JUST BELOW P0+180 DEGREES.        *
C     ******************************************************************
```

```
      IF (POBS .LT. PI) THEN
        PA=POBS+PI
      ELSE
        PA=POBS-PI
      END IF
      DO 3 M=1,NU-1
        IF ((PA .GE. U(M)) .AND. (PA .LT. U(M+1))) THEN
          IREF=M
        ELSE
        END IF
3     CONTINUE
C     ********************************************************************
C     * SEQUENCE THROUGH THE INDICES.                                    *
C     ********************************************************************
      M=IREF
      DO 4 L=1,NU-NS-1
        IS=(-1)**L
        M=M-IS*(L-1)
        IF (M .LT. 1) THEN
          M=M+NU-1
        ELSE IF (M .GT. NU-1) THEN
          M=M-(NU-1)
        ELSE
        END IF
        IF ((ABS(U(M)-POBS) .LT. PI/2.0)
     &                   .OR. (ABS(U(M)-POBS) .GT. 3.0*PI/2.0)) THEN
          IS=1
          NO=0
        ELSE
          IS=-1
          NO=NV
        END IF
        DO 5 K=1,NV-1
          N=NO+IS*K
C         ****************************************************************
C         * GENERATE THE RECTANGULAR POINTS.                             *
C         ****************************************************************
          CPI=COS(U(M))
          SPI=SIN(U(M))
          CPI1=COS(U(M+1))
          SPI1=SIN(U(M+1))
          XS1=V(N)*CPI
          YS1=V(N)*SPI
          ZS1=F(M,N)
          XS2=V(N)*CPI1
          YS2=V(N)*SPI1
          ZS2=F(M+1,N)
          XS3=V(N+1)*CPI1
          YS3=V(N+1)*SPI1
```

```
            ZS3=F(M+1,N+1)
            XS4=V(N+1)*CPI
            YS4=V(N+1)*SPI
            ZS4=F(M,N+1)
C          *************************************************************
C          * ROTATE THE 4 VERTICES OF THE QUADRILATERAL.               *
C          *************************************************************
            CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)
            CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
            CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
            CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
C          *************************************************************
C          * FILL THE QUADRILATERAL.                                   *
C          *************************************************************
            WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
C          *************************************************************
C          * DRAW PERIMETER OF THE QUADRILATERAL.                      *
C          *************************************************************
            WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
    5     CONTINUE
    4   CONTINUE
C     *****************************************************************
C     * SHOW THE PAGE.                                                *
C     *****************************************************************
        WRITE(2,100) 'showpage'
C     *****************************************************************
C     * FORMATS.                                                      *
C     *****************************************************************
  100   FORMAT(A72)
  101   FORMAT(F7.2,1X,F7.2,A58)
  200   FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l c')
  201   FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l w')
        END
C
        SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
        X=AX*COS(RP)*XA-AY*SIN(RP)*YA
        Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
        RETURN
        END
```

```
      PROGRAM CYLR3D
C     *********************************************************************
C     * THIS PROGRAM PRODUCES A POSTSCRIPT FILE REPRESENTING A THREE      *
C     * DIMENSIONAL PLOT OF A FUNCTION IN CYLINDRICAL COORDINATES.        *
C     *********************************************************************
C     * TIMOTHY J. PETERS                                LAST UPDATED     *
C     * THE AEROSPACE CORPORATION                        3/1/91           *
C     * 2350 EAST EL SEGUNDO BOULEVARD.                                   *
C     * EL SEGUNDO, CA 90245                                              *
C     *********************************************************************
C     * INPUTS:                                                           *
C     *                                                                   *
C     * NU   - NUMBER OF U POINTS.                                        *
C     * NV   - NUMBER OF V POINTS.                                        *
C     * AX  - X DIRECTION SCALE FACTOR.                                   *
C     * AY  - Y DIRECTION SCALE FACTOR.                                   *
C     * AZ  - Z DIRECTION SCALE FACTOR.                                   *
C     * PO  - PHI OBSERVATION ANGLE IN RANGE 0<=PO<=2PI.                  *
C     * TO  - THETA OBSERVATION ANGLE IN RANGE 0<=PO<=PI.                 *
C     * IC  - IF IC=1 THEN CONVERT THE FUNCTION VALUES TO DB.             *
C     *       NOTE THAT IF IC=1 THEN F MUST BE IN THE RANGE 0<=F<=1.      *
C     * ETA - PLOT FLOOR IN DB.                                           *
C     * NUU - EFFECTIVE ZERO IN DB.                                       *
C     * NS  - NUMBER OF SEGMENTS TO REMOVE.                               *
C     * U(MAX) - U COORDINATE ARRAY.                                      *
C     * V(MAX) - V COORDINATE ARRAY.                                      *
C     * F(MAX,MAX) - FUNCTION VALUE MATRIX.                               *
C     *                                                                   *
C     * OUTPUT:                                                           *
C     *                                                                   *
C     * CYLR.PS  - POSTSCRIPT FILE REPRESENTING THE PLOT.                 *
C     *                                                                   *
C     *********************************************************************
      PARAMETER (MAX=100)
      REAL*4 U(MAX),V(MAX),F(MAX,MAX),NUU
      OPEN(UNIT=2,FILE='CYLR.PS')
      REWIND(2)
      RAD=.17453293E-01
      PI=.3141593E+01
C     *********************************************************************
C     * READ THE INPUTS.                                                  *
C     *********************************************************************
      OPEN(UNIT=1,FILE='DATA')
      READ(1,*) NU,NV,AX,AY,AZ,PO,TO,IC,ETA,NUU,NS
      READ(1,*) (U(M),M=1,NU)
      READ(1,*) (V(N),N=1,NV)
      READ(1,*) ((F(M,N),M=1,NU),N=1,NV)
C     *********************************************************************
C     * DEFINE SOME MACROS IN POSTSCRIPT.                                 *
```

```
C     *******************************************************************
       WRITE(2,100) 'initgraphics erasepage letter'
       WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
       WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
       WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
C     *******************************************************************
C     * SET THE LINE CHARACTERISTICS.                                   *
C     *******************************************************************
       WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
C     *******************************************************************
C     * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAGE.       *
C     *******************************************************************
       XOFFSET=306.0
       YOFFSET=396.0
       WRITE(2,101) XOFFSET,YOFFSET,' t'
C     *******************************************************************
C     * CONVERT THE OBSERVATION ANGLES TO RADIANS.                      *
C     *******************************************************************
       POBS=RAD*PO
       TOBS=RAD*TO
C     *******************************************************************
C     * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION *
C     * ANGLES.                                                         *
C     *******************************************************************
       RP=-POBS-PI/2.0
       RT=TOBS
       CP=COS(RP)
       SP=SIN(RP)
       CT=COS(RT)
       ST=SIN(RT)
C     *******************************************************************
C     * IF REQUESTED CONVERT THE DATA TO DB SCALE.                      *
C     *******************************************************************
       IF (IC .EQ. 1) THEN
         TR=10.0**(0.1*ETA)
         DO 1 M=1,NU
          DO 2 N=1,NV
           IF(F(M,N) .LE. TR) THEN
             F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
           ELSE
             F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
           END IF
   2      CONTINUE
   1     CONTINUE
       ELSE
       END IF
C     *******************************************************************
C     * DETERMINE THE VALUE WHICH IS JUST BELOW PO+180 DEGREES.         *
C     *******************************************************************
```

```
      IF (POBS .LT. PI) THEN
        PA=POBS+PI
      ELSE
        PA=POBS-PI
      END IF
      DO 3 M=1,NU-1
        IF ((PA .GE. U(M)) .AND. (PA .LT. U(M+1))) THEN
          IREF=M
        ELSE
        END IF
3     CONTINUE
C     ********************************************************************
C     * SEQUENCE THROUGH THE INDICES.                                    *
C     ********************************************************************
      DO 4 K=1,NV-1
        N=K
        M=IREF
        DO 5 L=1,NU-NS-1
          IS=(-1)**L
          M=M-IS*(L-1)
          IF (M .LT. 1) THEN
            M=M+NU-1
          ELSE IF (M .GT. NU-1) THEN
            M=M-(NU-1)
          ELSE
          END IF
C         ****************************************************************
C         * GENERATE THE RECTANGULAR POINTS.                             *
C         ****************************************************************
          CPI=COS(U(M))
          SPI=SIN(U(M))
          CPI1=COS(U(M+1))
          SPI1=SIN(U(M+1))
          XS1=F(M,N)*CPI
          YS1=F(M,N)*SPI
          ZS1=V(N)
          XS2=F(M+1,N)*CPI1
          YS2=F(M+1,N)*SPI1
          ZS2=V(N)
          XS3=F(M+1,N+1)*CPI1
          YS3=F(M+1,N+1)*SPI1
          ZS3=V(N+1)
          XS4=F(M,N+1)*CPI
          YS4=F(M,N+1)*SPI
          ZS4=V(N+1)
C         ****************************************************************
C         * ROTATE THE POINTS.                                           *
C         ****************************************************************
          CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)
```

```
         CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
         CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
         CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
C        ****************************************************************
C        * FILL THE QUADRILATERAL.                                      *
C        ****************************************************************
         WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
C        ****************************************************************
C        * DRAW PERIMETER OF THE QUADRILATERAL.                         *
C        ****************************************************************
         WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
   5   CONTINUE
   4  CONTINUE
C    ********************************************************************
C    * SHOW THE PAGE.                                                   *
C    ********************************************************************
      WRITE(2,100) 'showpage'
C    ********************************************************************
C    * FORMATS                                                          *
C    ********************************************************************
 100  FORMAT(A72)
 101  FORMAT(F7.2,1X,F7.2,A58)
 200  FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l c')
 201  FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l w')
      END
C
      SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
      X=AX*COS(RP)*XA-AY*SIN(RP)*YA
      Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
      RETURN
      END
```

```
      PROGRAM SPHR3D
C    ********************************************************************
C    * THIS PROGRAM PRODUCES A POSTSCRIPT FILE REPRESENTING A THREE     *
C    * DIMENSIONAL PLOT OF A FUNCTION IN SPHERICAL COORDINATES.         *
C    ********************************************************************
C    * TIMOTHY J. PETERS                               LAST UPDATED     *
C    * THE AEROSPACE CORPORATION                       3/1/91           *
C    * 2350 EAST EL SEGUNDO BOULEVARD.                                  *
C    * EL SEGUNDO, CA 90245                                             *
C    ********************************************************************
C    * INPUTS:                                                          *
C    *                                                                  *
C    * NU   - NUMBER OF U POINTS.                                       *
C    * NV   - NUMBER OF V POINTS.                                       *
C    * U(MAX) - U COORDINATE ARRAY.                                     *
C    * V(MAX) - V COORDINATE ARRAY.                                     *
C    * F(MAX,MAX) - FUNCTION VALUE MATRIX.                              *
C    * PO  - PHI OBSERVATION ANGLE IN RANGE 0<=PO<=2PI.                 *
C    * TO  - THETA OBSERVATION ANGLE IN RANGE 0<=PO<=PI.                *
C    * IC  - IF IC=1 THEN CONVERT THE FUNCTION VALUES TO DB.            *
C    *       NOTE THAT IF IC=1 THEN F MUST BE IN THE RANGE 0<=F<=1.     *
C    * ETA - PLOT FLOOR IN DB.                                          *
C    * NUU - EFFECTIVE ZERO IN DB.                                      *
C    * NS  - NUMBER OF SEGMENTS TO REMOVE.                              *
C    * AX  - X DIRECTION SCALE FACTOR.                                  *
C    * AY  - Y DIRECTION SCALE FACTOR.                                  *
C    * AZ  - Z DIRECTION SCALE FACTOR.                                  *
C    *                                                                  *
C    * OUTPUT:                                                          *
C    *                                                                  *
C    * SPHR.PS  - POSTSCRIPT FILE REPRESENTING THE PLOT.                *
C    *                                                                  *
C    ********************************************************************
      PARAMETER (MAX=100)
      REAL*4 U(MAX),V(MAX),F(MAX,MAX),NUU
      INTEGER P,Q
      OPEN(UNIT=2,FILE='SPH.PS')
      REWIND(2)
      RAD=.17453293E-01
      PI=.3141593E+01
C    ********************************************************************
C    * READ THE INPUTS.                                                 *
C    ********************************************************************
      OPEN(UNIT=1,FILE='DATA')
      READ(1,*) NU,NV,AX,AY,AZ,PO,TO,IC,ETA,NUU,NS
      READ(1,*) (U(M),M=1,NU)
      READ(1,*) (V(N),N=1,NV)
      READ(1,*) ((F(M,N),M=1,NU),N=1,NV)
C    ********************************************************************
```

```
C     * DEFINE SOME MACROS IN POSTSCRIPT.                                  *
C     ********************************************************************
       WRITE(2,100) 'initgraphics erasepage letter'
       WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
       WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
       WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
C     ********************************************************************
C     * SET THE LINE CHARACTERISTICS.                                      *
C     ********************************************************************
       WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
C     ********************************************************************
C     * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAPER.         *
C     ********************************************************************
       XOFFSET=306.0
       YOFFSET=396.0
       WRITE(2,101) XOFFSET,YOFFSET,' t'
C     ********************************************************************
C     * CONVERT THE OBSERVATION ANGLES TO RADIANS.                         *
C     ********************************************************************
       POBS=RAD*U0
       TOBS=RAD*V0
C     ********************************************************************
C     * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION *
C     * ANGLES.                                                            *
C     ********************************************************************
       ALPHA=-PI/2.0-POBS
       BETA=TOBS
C     ********************************************************************
C     * IF REQUESTED CONVERT THE DATA TO DB SCALE.                         *
C     ********************************************************************
       IF (IC .EQ. 1) THEN
         TR=10.0**(0.1*ETA)
         DO 1 M=1,NU
          DO 2 N=1,NV
           IF(F(M,N) .LE. TR) THEN
             F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
           ELSE
             F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
           END IF
   2      CONTINUE
   1     CONTINUE
       ELSE
       END IF
C     ********************************************************************
C     * DETERMINE THE INDEX OF THE ANGLES PHI_0 AND PHI_A                  *
C     ********************************************************************
       IF (POBS .LT. PI) THEN
         PA=POBS+PI
       ELSE
```

```fortran
          PA=POBS-PI
        END IF
        DO 3 K=1,NU-1
          IF ((POBS .GE. U(K)) .AND. (POBS .LE. U(K+1))) THEN
            MO=K
          ELSE
          END IF
          IF ((PA .GE. U(K)) .AND. (PA .LT. U(K+1))) THEN
            MA=K
          ELSE
          END IF
    3   CONTINUE
C      ******************************************************************
C      * DETERMINE THE INDEX OF THE ANGLE TOBS.                         *
C      ******************************************************************
        TA=PI-TOBS
        NO=NV
        NA=NV
        DO 4 K=1,NV-1
          IF ((TOBS .GE. V(K)) .AND. (TOBS .LT. V(K+1))) THEN
            NO=K
          ELSE
          END IF
          IF ((TA .GE. V(K)) .AND. (TA .LT. V(K+1))) THEN
            NA=K
          ELSE
          END IF
    4   CONTINUE
C      ******************************************************************
C      * BEGIN MAIN LOOP.                                               *
C      ******************************************************************
        M=MA
        DO 5 L=1,NU-NS-1
          M=M-((-1)**L)*(L-1)
          IF (M .LT. 1) THEN
            M=M+NU-1
          ELSE IF (M .GT. NU-1) THEN
            M=M-(NU-1)
          ELSE
          END IF
C        ****************************************************************
C        * SET THE CONSTANTS FOR DRAWING V.                             *
C        ****************************************************************
          DP=U(M)-POBS
          IF ((ABS(DP) .LT. PI/2.0) .OR. (ABS(DP) .GT. 3.0*PI/2.0)) THEN
            I=NV
            J=0
            L1=NV-NO
            L2=NO-1
```

```
      ELSE
        I=NA
        J=NA-1
        L1=NA-1
        L2=NV-NA
      END IF
C     ****************************************************************
C     * DRAW THE V DEPENDENCY IN THE FIRST DIRECTION.                *
C     ****************************************************************
      DO 6 K=1,L1
        N=I-K
C       **************************************************************
C       * COMPUTE THE 4 VERTICES OF THE FIRST QUADRILATERAL.         *
C       **************************************************************
        CPA=COS(U(M))
        SPA=SIN(U(M))
        CPB=COS(U(M+1))
        SPB=SIN(U(M+1))
        CQA=COS(V(N))
        SQA=SIN(V(N))
        CQB=COS(V(N+1))
        SQB=SIN(V(N+1))
C       **************************************************************
C       * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL.               *
C       **************************************************************
        C=F(M,N)*SQA
        XS1=C*CPA
        YS1=C*SPA
        ZS1=F(M,N)*CQA
        C=F(M+1,N)*SQA
        XS2=C*CPB
        YS2=C*SPB
        ZS2=F(M+1,N)*CQA
        C=F(M+1,N+1)*SQB
        XS3=C*CPB
        YS3=C*SPB
        ZS3=F(M+1,N+1)*CQB
        C=F(M,N+1)*SQB
        XS4=C*CPA
        YS4=C*SPA
        ZS4=F(M,N+1)*CQB
C       **************************************************************
C       * ROTATE THE 4 VERTICES OF THE QUADRILATERAL.                *
C       **************************************************************
        CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,ALPHA,BETA)
        CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,ALPHA,BETA)
        CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,ALPHA,BETA)
        CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,ALPHA,BETA)
C       **************************************************************
```

```
C        * FILL THE QUADRILATERAL.                                            *
C        **********************************************************************
          WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
C        **********************************************************************
C        * DRAW PERIMETER OF THE QUADRILATERAL.                               *
C        **********************************************************************
          WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
    6   CONTINUE
C      ************************************************************************
C      * DRAW THE V DEPENDENCY IN THE SECOND DIRECTION.                       *
C      ************************************************************************
        DO 7 K=1,L2
          N=J+K
C        **********************************************************************
C        * COMPUTE THE 4 VERTICES OF THE FIRST QUADRILATERAL.                 *
C        **********************************************************************
          CPA=COS(U(M))
          SPA=SIN(U(M))
          CPB=COS(U(M+1))
          SPB=SIN(U(M+1))
          CQA=COS(V(N))
          SQA=SIN(V(N))
          CQB=COS(V(N+1))
          SQB=SIN(V(N+1))
C        **********************************************************************
C        * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL.                       *
C        **********************************************************************
          C=F(M,N)*SQA
          XS1=C*CPA
          YS1=C*SPA
          ZS1=F(M,N)*CQA
          C=F(M+1,N)*SQA
          XS2=C*CPB
          YS2=C*SPB
          ZS2=F(M+1,N)*CQA
          C=F(M+1,N+1)*SQB
          XS3=C*CPB
          YS3=C*SPB
          ZS3=F(M+1,N+1)*CQB
          C=F(M,N+1)*SQB
          XS4=C*CPA
          YS4=C*SPA
          ZS4=F(M,N+1)*CQB
C        **********************************************************************
C        * ROTATE THE 4 VERTICES OF THE QUADRILATERAL.                        *
C        **********************************************************************
          CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,ALPHA,BETA)
          CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,ALPHA,BETA)
          CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,ALPHA,BETA)
```

```
      CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,ALPHA,BETA)
C     ***************************************************************
C     * FILL THE QUADRILATERAL.                                     *
C     ***************************************************************
      WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
C     ***************************************************************
C     * DRAW PERIMETER OF THE QUADRILATERAL.                        *
C     ***************************************************************
      WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
  7  CONTINUE
C    *****************************************************************
C    * CHECK AND SEE IF A CUT IS REQUESTED.                          *
C    *****************************************************************
     IF ((NS .GT. 0) .AND. (L .GT. NU-NS-3)) THEN
C     ***************************************************************
C     * GENERATE A POLYGON IN THETA AT A CONSTANT PHI.              *
C     ***************************************************************
      IF (POBS .LE. PI) THEN
        IF ((U(M) .GE. POBS).AND.(U(M) .LE. PA)) THEN
          P=M
        ELSE
          P=M+1
        END IF
      ELSE
        IF ((U(M) .GE. PA).AND.(U(M) .LE. POBS)) THEN
          P=M+1
        ELSE
          P=M
        END IF
      END IF
      CPA=COS(U(P))
      SPA=SIN(U(P))
      CQA=COS(V(1))
      SQA=SIN(V(1))
      C=F(P,1)*SQA
      XS1=C*CPA
      YS1=C*SPA
      ZS1=F(P,1)*CQA
      CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,ALPHA,BETA)
      WRITE(2,300) X1,Y1
      DO 8 Q=1,NV
        CQA=COS(V(Q))
        SQA=SIN(V(Q))
        C=F(P,Q)*SQA
        XS1=C*CPA
        YS1=C*SPA
        ZS1=F(P,Q)*CQA
        CALL ROTATE(XS1,YS1,ZS1,XQ,YQ,AX,AY,AZ,ALPHA,BETA)
        WRITE(2,301) XQ,YQ
```

```
    8       CONTINUE
            WRITE(2,100) ' c '
C         ************************************************************
C         * DRAW A LINE AROUND THE PERIMETER OF THE POLYGON.         *
C         ************************************************************
            WRITE(2,100) ' newpath '
            WRITE(2,100) ' 0 sg '
            WRITE(2,300) X1,Y1
            DO 9 Q=1,NV
              CQA=COS(V(Q))
              SQA=SIN(V(Q))
              C=F(P,Q)*SQA
              XS1=C*CPA
              YS1=C*SPA
              ZS1=F(P,Q)*CQA
              CALL ROTATE(XS1,YS1,ZS1,XQ,YQ,AX,AY,AZ,ALPHA,BETA)
              WRITE(2,301) XQ,YQ
    9       CONTINUE
            WRITE(2,100) ' s '
          ELSE
          END IF
    5   CONTINUE
C     ****************************************************************
C     * SHOW THE PAGE.                                               *
C     ****************************************************************
        WRITE(2,100) 'showpage'
C     ****************************************************************
C     * FORMATS.                                                     *
C     ****************************************************************
  100   FORMAT(A72)
  101   FORMAT(F7.2,1X,F7.2,A58)
  200   FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l c')
  201   FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
     &F7.2,' l ',F7.2,1X,F7.2,' l w')
  300   FORMAT(F7.2,1X,F7.2,' m')
  301   FORMAT(F7.2,1X,F7.2,' l')
        END
C
        SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
        X=AX*COS(RP)*XA-AY*SIN(RP)*YA
        Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
        RETURN
        END
```